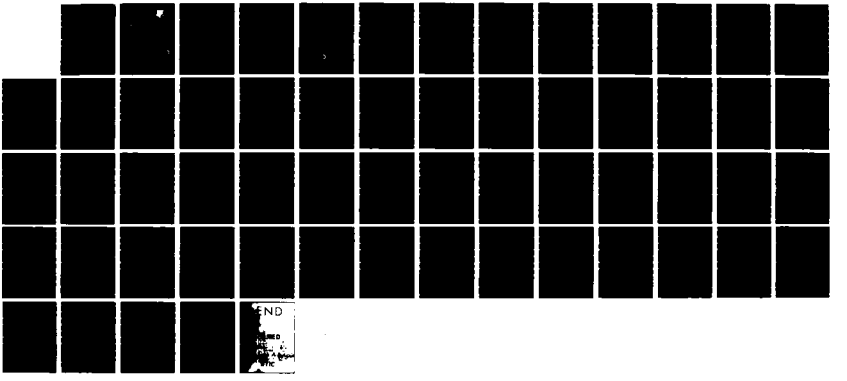
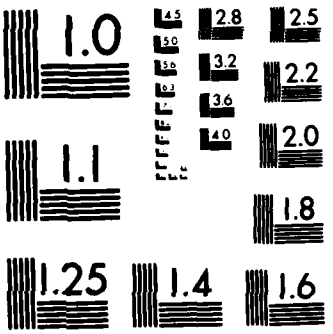
AD-A146 844 SOFTWARE TEST HANDBOOK(U) BOEING AEROSPACE CO SEATTLE WA ENGINEERING TECHNOLOGY DIV E PRESSON MAR 84 RADC-TR-84-53-VOL-1 F30602-82-C-0059 1/1

UNCLASSIFIED F/G 9/2 NL





12

RADC-TR-84-53, Vol I (of two)
Final Technical Report
March 1984

SOFTWARE TEST HANDBOOK

Boeing Aerospace Company

Edward Presson

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-84-53, Vol I (of two) has been reviewed and is approved for publication.

APPROVED: Frank S LaMonica

FRANK S. LaMONICA
Project Engineer

APPROVED: Raymond P. Urtz Jr.

RAYMOND P. URTZ, JR.
Acting Technical Director
Command & Control Division

FOR THE COMMANDER: John A. Ritz

JOHN A. RITZ
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COEE) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A	5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-84-53, Vol I (of two)
6a. NAME OF PERFORMING ORGANIZATION Boeing Aerospace Company Engineering Technology 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (COEE)
6c. ADDRESS (City, State and ZIP Code) Seattle WA 98124	7b. ADDRESS (City, State and ZIP Code) Griffiss AFB NY 13441
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center 8b. OFFICE SYMBOL (If applicable) COEE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-82-C-0059
8c. ADDRESS (City, State and ZIP Code) Griffiss AFB NY 13441	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. 63728F; PROJECT NO. 2527; TASK NO. 02; WORK UNIT NO. 09
11. TITLE (Include Security Classification) SOFTWARE TEST HANDBOOK	
12. PERSONAL AUTHOR(S) Edward Presson	
13a. TYPE OF REPORT Final 13b. TIME COVERED FROM Mar 82 TO Sep 83	14. DATE OF REPORT (Yr., Mo., Day) March 1984 15. PAGE COUNT 60
16. SUPPLEMENTARY NOTATION N/A	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
09	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Testing
Software
Computer Programs

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of the Software Test Handbook effort was to provide Air Force software developers with guidelines and methodology for the effective use of higher order language (HOL) software testing techniques and in the selection of automated tools for the testing of computer programs.

The effort resulted in a two volume final technical report. This report, Volume I - Final Technical Report, describes the total contractual effort including a project overview, summary for each of three technical tasks, and a bibliography. Volume II, the Software Test Guidebook, contains the guidelines and methodology resulting from the effort.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ OTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Frank S. LaMonica	22b. TELEPHONE NUMBER (Include Area Code) (315) 330-3977 22c. OFFICE SYMBOL RADC (COEE)

PREFACE

This document is the final technical report, CDRL item A005, that describes the results of the three tasks involved in developing the Software Test Guidebook. This report was prepared in accordance with the statement of work, contract F30602-82-C-0059. It summarizes the activities performed for the Rome Air Development Center by Boeing Aerospace Company in Kent, Washington.

In task 1 of this contract, a survey was made to determine current software testing practices used in the five major Air Force missions. The survey was supplemented with five site visits. Task 2 evaluated current state-of-the-art software testing techniques and test tools and incorporated this information into tables. Task 3 designed and prepared the handbook.

This report is in two volumes: the technical report that describes the total contractual effort and the Software Test Guidebook that describes the methodology for selecting testing techniques and test tools. The technical report comprises a project overview in section 1.0; summaries of tasks 1, 2, and 3 in sections 2.0, 3.0, and 4.0; and a bibliography in section 5.0. The Software Test Guidebook is divided into six sections. It is designed to assist Air Force software developers in using higher order language software testing techniques and in selecting automated tools to test computer programs.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AD	Armament Division
ADC	Aerospace Defense Center
ADP	automatic data processing
AFTI	advanced fighter technology integration
ALCM	air-launched cruise missile
ASD	Aeronautical Systems Division
ATE	automatic test equipment
ATO	Air Tasking Order
BITS	built-in test
CAFMS	Computer Assisted Air Force Management System
CCPDS	Command Center Processing and Display System
CINCSAC	Commander-in-Chief Strategic Air Command
CITS	central integrated test systems
COM	computer output microfiche
CPCI	computer program configuration item
CPU	central processor unit
C3	command, control, and communications
CSOC	Consolidated Space Operations Center
DGZ	designated ground zero
DT&E	development, test and evaluation
ELINT	electronic intelligence
HDM	hierarchical design methodology
HOL	higher order language
ICBM	intercontinental ballistic missile
IDHS	intelligence data handling system
IUS	inertial upper stage
IV&V	independent verification and validation

JINTACCS	Joint Interoperable Tactical Air Command and Control System
JSCS	Joint Strategic Connectivity Staff
JSTPS	Joint Strategic Target Planning Staff
LRU	line replaceable unit
MATE	Modular Automatic Test Equipment (program)
NMCC	National Military Command Center
O&S	operations and support
OFP	operational flight programs
OPR	Office of Primary Responsibility
OT&E	operational test and evaluation
PROM	programmable read-only memory
SAC	Strategic Air Command
SCF	Satellite Control Facility
SD	Space Division
SDL	software development laboratory
SILTF	System Integration Laboratory and Test Facility
SIOP	Single Integrated Operational Plan
SLBM	submarine-launched ballistic missile
SOLARS	SAC On-Line Analysis and Retrieval System
SPO	System Program Office
SREM/REVS	Software Requirements Engineering Methodology/Requirement Engineering Validation System
SRU	shop replaceable unit
TAC	Tactical Air Command
TACC	Tactical Air Control Center
TACS	Tactical Air Control System
TRD	Test Requirements Document
TRICOMS	Triad Computer System

TTY	teletypewriter
USAF	United States Air Force
UUT	unit under test
V&V	verification and validation
WWMCCS	Worldwide Military Command and Control System

1.0 PROJECT OVERVIEW

1.1 BACKGROUND

The testing and operational support of computer programs continue to be critical information processing problems facing the Air Force. Substantial resources in terms of funds and personnel are continually applied during the software development life cycle for testing and maintaining computer programs. However, in many instances the application of these resources does not achieve the desired result; that is, programs thought to be correct may suddenly produce erroneous outputs during operational use.

Traditionally, testing, verification, and validation of software have been a largely manual process. For example, test drivers and data are usually prepared manually and the test results manually interpreted. A wide variety of additional software testing and verification techniques has been developed in recent years, many of which can and have been implemented in automated software tools. While the effectiveness of manually applied techniques can vary considerably with the skill with which they are applied, many software tools can be highly consistent and reliable in their ability to detect the errors for which they were designed. These software tools can potentially improve the quality of testing and at the same time reduce the manual effort required to test a software system.

Typical testing activities that occur in the software development life cycle include unit testing, component testing, integration testing, system-level and acceptance testing, and maintenance testing (retesting). These activities, which may differ slightly from one development environment to another, are each associated with different objectives and impose specialized requirements on the testing task. As a result, certain testing techniques may be more applicable to one activity than to another.

In many cases, while the techniques implemented by the tools have proved effective, the use of these tools in military software development and support environments has been lacking. In part, the low utilization is because managers and engineers are not well informed about the availability of tools and techniques, their usefulness and effectiveness, and how they can be integrated properly into specific development and support environments. While a number of regulations and guidebooks for software development have been

prepared in the past, most have dealt with providing an understanding of the software development life cycle, and little emphasis has been placed on using software testing tools and techniques in the life cycle.

1.2 PROJECT SUMMARY

The purpose of the Software Test Guidebook project is to provide Air Force software developers with a guidebook to guide them in the effective use of higher order language (HOL) software testing techniques and in the selection of automated tools for the testing of computer programs. The guidebook specifies guidelines and methodologies for understanding and applying automated state-of-the-art testing techniques in various types of Air Force software development and support environments. This effort will foster the transfer of advanced software testing technology to the Air Force user community. Air Force software developers may find the results of this effort useful in supplementing existing regulations and guidebooks.

1.3 SCOPE OF EFFORT

Guidelines and methodologies were developed that describe the proper use of advanced software testing technology during the development of computer application software for the five primary Air Force missions (armament, avionics, command, control, and communications (C^3), missile/space, and mission/force management). The guidelines and methodologies pertain to those testing activities of the software development life cycle that follow the beginning of actual program coding. Representative Air Force software sites were visited and analyzed to determine typical characteristics of application software and environments in which application software is developed and maintained. Characteristics of advanced state-of-the-art software testing technology were extracted from the literature and from available software tool surveys. The guidelines and methodologies developed have been provided in the form of a guidebook.

Guidelines and methodologies were developed for the selection of state-of-the-art software testing techniques in the computer program development life cycle; that is, development test and evaluation (DT&E), operational test and evaluation (OT&E), and verification and validation (V&V), as defined in AFR 80-14 and AFR 800-14, with the following constraints:

a. This effort covered only the coding and checkout phase, test and integration phase, and operation and support (O&S) phase. Further, for the O&S phase, the investigation was limited to the support aspects (i.e., coding, checkout, and retesting of modifications to deployed computer programs).

b. This effort considered not only those testing techniques pertinent to the development of operational computer programs for the five primary Air Force missions, but also the use of those testing techniques in the development of auxiliary software programs (e.g., simulators and data analysis programs) used to test and support the operational computer programs.

1.4 Brief Description of the Guidebook

The principal purpose of the guidebook is to select state-of-the-art testing techniques for organic Air Force software testing. In addition, the guidebook can be used for preparation of a Statement of Work and for evaluation of proposals. All three applications use the same table-driven methodology to determine appropriate software testing techniques.

The guidebook is designed to permit a user to determine appropriate software test techniques, starting with the knowledge of which USAF mission is being supported. Using the unique guidebook appendix to his mission, the user can identify the specific software types to be tested. With this knowledge, the user will be led to the generic software type corresponding to the software function at hand. With the generic software type (from the guidebook appendix) and testing confidence level, the user can go to another table to find the appropriate software testing techniques for his unique situation.

The table-driven methodology provides three paths for technique selection. The first path is based on the kind of software being tested and the testing confidence level required. The second path is based on the test phases and test objectives, and the third path is based on detection of specific software error types.

The contract was titled "Software Test Handbook," but "Handbook" was changed to "Guidebook" to better reflect the purpose, which was to assist in the selection of testing techniques, not to rigidly define them.

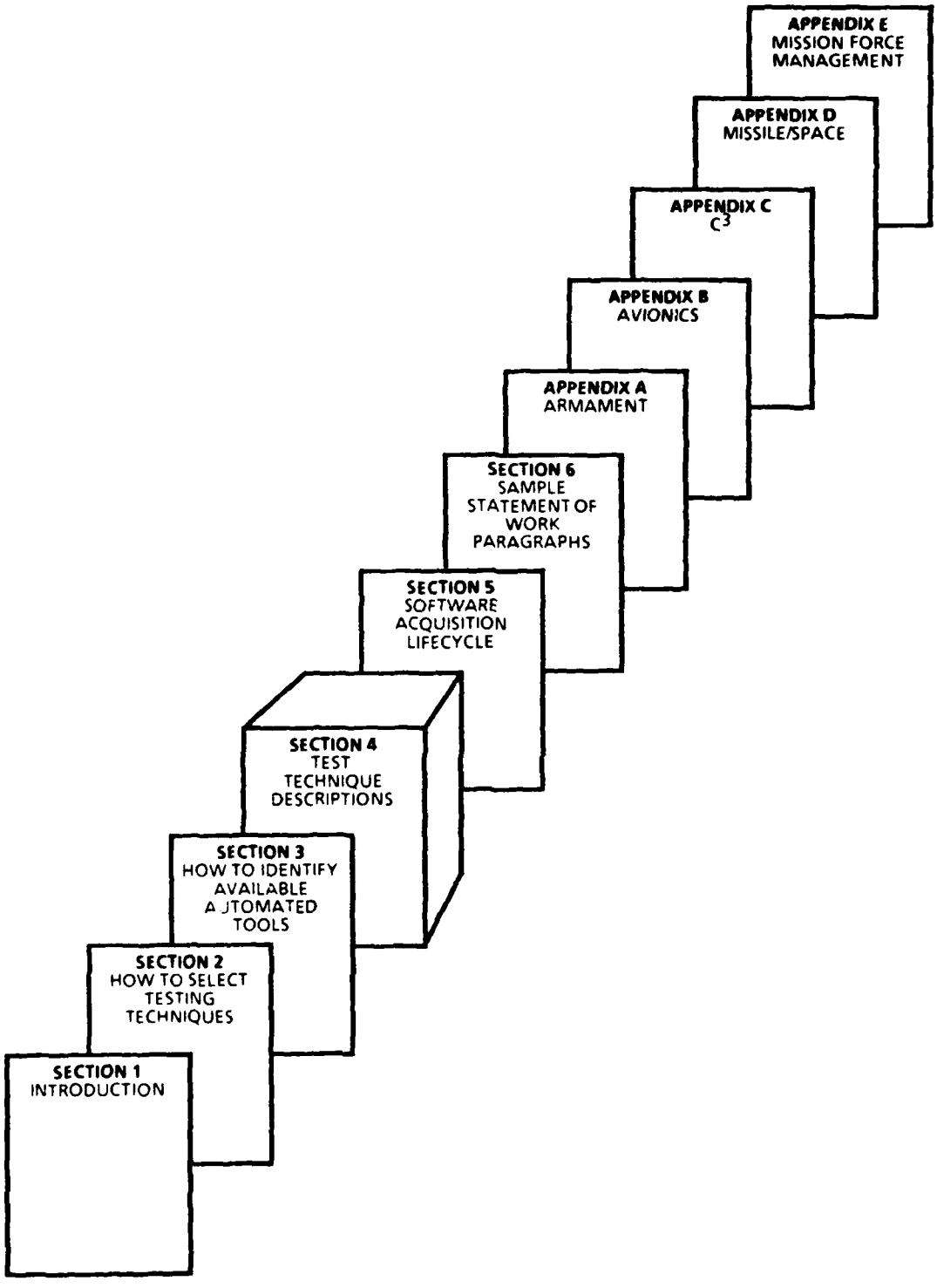


Figure 1. Guidebook Organization

The Software Test Guidebook comprises six major sections and five appendices, as shown in figure 1. A brief summary of its contents is found in the following paragraphs.

Section 1.0 states the objectives of the guidebook, describes its outline and content, and discusses its applications.

Section 2.0 presents a compact set of instructions, guidelines, and tables for selecting software testing techniques. It also includes sections on the selection of software support tools and on test completion criteria.

Section 3.0 contains a list of the major catalogs that provide information on automated software tools. It has tables that provide a cross reference to three tool catalogs for determining the availability of existing software tools that support the techniques selected by the guidebook user.

Section 4.0 defines the terms used in the taxonomy of testing techniques and gives a detailed description of state-of-the-art testing techniques. These descriptions discuss the technique and related considerations such as cost, user training, and hardware requirements.

Section 5.0 discusses the software life cycle, software acquisition cycle, and the normal phases of testing, as defined in AFR 80-14 and 800-14. This section is a supplement to the guidebook and does not replace the Air Force regulations.

Section 6.0 has several model statement of work (SOW) paragraphs that may be used as prototypes by Air Force acquisition managers in preparing a request for proposal (RFP).

The appendices describe five Air Force mission areas: armament; avionics; command, control, and communication (C^3); missile/space; and mission/force management. Each appendix lists software functions characteristic of the computer programs developed within that mission. These functions are each assigned a software category number.

1.5 OUTLINE OF INVESTIGATION

The Software Test Guidebook development comprised three major tasks, which are outlined in sections 1.5.1 through 1.5.3.

1.5.1 Task 1, Programming Environment Survey

An investigation to analyze programming characteristics and the development and support environments of typical application software for the five primary Air Force missions was conducted in task 1.

Within each primary mission, differences in the application software that impact the most appropriate testing and verification strategy were investigated. The following characteristics were addressed:

- Operating instructions and regulations used to standardize programming and testing methods and to document the testing process.
- Level of robustness (fault tolerance).
- Timing and synchronization requirements; real-time processing constraints.
- Distributed and centralized processing configurations.
- Application program processing requirements.
- Type and level of HOL programming languages used.
- Level of support and testing provided by compilers.
- Software testing tools and techniques and debugging aids typically used.
- Code complexity (data structures, control structures, data types).
- Real and non-real-time applications.
- Where coding and testing are performed (type of machine).
- Availability and abundance of memory, central processor unit (CPU), and disk storage.
- Availability of input-output (I/O) to an external storage media.
- Computer host and target relationship.
- Use and non-use of environment simulators on host computer.
- Batch and interactive mode of use.
- Size and number of modules in a typical application and sizing criticality.
- Evolutionary requirements of a typical software application.

- Type and criticality of software application (or its components).
- Mission-imposed time constraints for software modification.
- Testing requirements included in statements-of-work for contracted software development.

To collect characteristic data representative of the applications as specified in the preceding list, the following Air Force software development sites were visited.

- Tactical Air Command Headquarters, TAC/ADY, Langley AFB, Virginia.
- Aeronautical Systems Division, ASD/EN, Wright-Patterson AFB, Ohio.
- Space Command, SPACECOM/KRS, Peterson AFB, Colorado.
- Space Division, SD/AGM, Los Angeles AFS, California.
- Strategic Air Command Headquarters, SAC/AD, Offutt AFB, Nebraska.
- Armament Division, AD/SDEE, Eglin AFB, Florida.

A questionnaire was prepared and distributed to each of the designated sites prior to the scheduled visit. It addressed collecting current data pertinent to the previously listed application software and development environment characteristics. Results of this task were reported in the task 1 Interim Technical Report, CDRL item A003, dated February 1983, and are summarized in section 2.0. These results were also used to prepare appendices A through E of the Software Test Guidebook to describe the five primary Air Force mission areas. The P/M Group (1546 Marsetta Drive, Dayton, Ohio 45432) was responsible for the survey of AD and ASD.

1.5.2 Task 2, Evaluation of Testing Techniques

State-of-the-art HOL software testing techniques most appropriate to the individual testing and verification phases of each type of Air Force application software and its associated development and support environment were determined. A correlation of test requirements with those techniques most suited for verifying them was provided.

Examples of such requirements include module interface checking, timing and synchronization, and fault tolerance.

To determine the applicability of testing techniques to particular Air Force environments, the following characteristics were considered:

a. Performance analysis capability.
b. Error detection and location capability.
c. Side effects and benefits (e.g., computer program documentation).
d. Cost, schedule, and benefits impact.
e. Management impact.
f. Training required (user and maintenance training).
g. Usage constraints (machine, language, and operating system).
h. Typical computer resources required (e.g., memory and CPU time).
i. Level of human interaction required.
j. Usefulness of technique in supporting modern programming practices such as structured programming and structured walkthroughs.

Software tool taxonomies and surveys were used to define and identify software testing tools and techniques, and their typical performance characteristics. Results of this task are reported in section 3.0.

1.5.3 Task 3, Preparation of Software Test Guidebook

Guidelines and methodologies for using the applicable techniques during DT&E, OT&E, and V&V for each Air Force mission area were developed and compiled into a guidebook. The organization of the guidebook was designed around the table-driven methodology. The design goal was to produce a guidebook that would be easy to use for both the first-time reader as well as the experienced guidebook user. To achieve this goal, the guidebook was organized so that its use would appear intuitively right to the new reader, but arranged so the experienced user could accomplish his task efficiently. The guidebook organization and usage was illustrated with a series of figures to further clarify the structure. The text was written by software engineers and reviewed by a technical editor; this process went through several iterations. Additionally, the contract technical monitor provided many comments on early drafts as the guidebook evolved.

2.0 SUMMARY OF TASK I

2.1 CHRONOLOGY OF ACTIVITIES

The following list of activities were completed in task 1.

a. The Software Test Handbook contract was awarded on March 3, 1982.

b. A kickoff meeting was held at RADC in Rome, New York, on March 25, 1982, where plans for the task I activities were discussed and a detailed proposal was presented for the questionnaire outline, scope, and coverage.

c. Air Force site representatives were contacted by telephone to familiarize them with the survey process; this activity took place during the week of April 17, 1982.

d. A pilot survey was conducted in May 1982, using the ASAT Missile Guidance Computer Program. This pilot survey was followed by interviews of the individuals who completed questionnaire forms in order to determine the time required to complete questionnaire sections and identify any deficiencies. Comments received from the pilot survey were incorporated into a revision to the questionnaire.

e. The draft questionnaire form was distributed to eight Air Force site representatives for their review and evaluation on May 14, 1982. Completion of this review required 1 month.

f. The questionnaires were completed by the following Air Force sites and returned by September 1982.

   1. Armament Division, AD/ENEC, Eglin AFB, Florida.

   2. Aerospace Defense Center, ADC/KRS, Peterson AFB, Colorado.

   3. Aeronautical Systems Division, ASD/EN, Wright-Patterson AFB, Ohio.

   4. Space Division, SD/AGM, Los Angeles AFS, California.

   5. Tactical Air Command, TAC/ADY, Langley AFB, Virginia.

g. Air Force onsite survey visits were conducted by Boeing and P/M Group personnel at six sites on the following dates:

1. Space Division (SD), Los Angeles. September 15 through 19, 1982, (P/M Group).

2. Aeronautical Systems Division (ASD), Wright-Patterson AFB. September 20 through 23, 1982, (P/M Group).

3. Aerospace Defense Center (ADC), Peterson AFB. September 20 through 21, 1982, (Boeing).

4. Tactical Air Command (TAC), Langley AFB. September 23 through 24, 1982, (Boeing).

5. Strategic Air Command (SAC), Offutt AFB. October 4 through 5, 1982, (Boeing).

6. Armament Division (AD), Eglin AFB. October 7 through 8, 1982, (Boeing).

h. At the four sites surveyed by Boeing, briefings were presented to Air Force personnel from all participating organizations. The presentations covered the purpose of the guidebook, the approach for using the guidebook, examples of some of its tables, and a taxonomy for software test techniques and tools developed for the guidebook.

i. Air Force site summaries were prepared during November 1982. These summaries were derived from a composite of the data obtained from the questionnaire responses and onsite visits.

j. An oral presentation covering a summary of task I activities took place at RADC on February 1, 1983. Representatives of all participating Air Force organizations were invited by RADC to attend this review.

2.2 SURVEY METHODOLOGY

The scope of the survey was well defined by the statement of work for the contract. The tasks to implement the survey required defining a general strategy for effectively obtaining the required information from participating sites. An underlying assumption was that the principal problems would be (1) limited availability of personnel to provide survey data and (2) determination of how to select the target software for survey coverage, so that the results would provide representative data; survey forms for recording data must be compatible with the level of the software. It would have been desirable to select classes of software for conducting the survey for each site. However, this was not practical because (1) the classes at the various sites were not known and (2) classification of software for site and mission areas was considered a necessary product derived from the survey, rather than an input to it.

It was concluded that the survey questionnaires should be directed toward individual computer program configuration items (CPCI). Each site representative was requested to select between four and eight computer programs, typical of their software, for the survey. This number was a compromise between a desire for a more statistically valid sampling and the significant amount of time needed to complete the questionnaires. The pilot survey participants reported that it took them approximately 3.5 hours to fill out one questionnaire.

Another consideration in formulating the questionnaire was whether the basic information should be obtained in person-to-person interviews or by a mailed-in form, followed up by a more generalized interview covering the total site software development and maintenance environment. It was concluded that the latter alternative would be a more efficient use of contract funds, permit greater flexibility in the nature of the onsite interviews, and help minimize bias.

Once the basic approach for the survey was defined—a stand alone questionnaire, followed by an unencumbered site visit—it was necessary to establish the format of the questionnaire to provide the needed information. Two general approaches were considered: one emphasized essay questions, the other emphasized multiple-choice questions. It was concluded that essay questions would be most desirable for person-to-person interviews, while multiple choice would be best for the mail-in-form approach. Therefore,

multiple-choice questions were selected for the questionnaire, despite the fact that they would be more difficult to prepare. Essay questions were considered too unstructured for the survey.

The questionnaire form was divided into six independent sections. The rationale for this was twofold: (1) to encourage qualified personnel to provide information in areas of their expertise and (2) to provide the capability to distribute the effort among several participants, rather than encumbering a single person. One large questionnaire was considered more likely to be deferred in the work scheduling than a sectionalized one. Many of the questionnaires were completed by more than one person, but rarely by more than three. The six sections of the questionnaire were as follows:

a. Part 1, General Software Information. Acquires information concerning the size and the technical and contractual requirements for a computer program. This part was intended to be completed by individuals such as program managers or administrative officers with project-level visibility.

b. Part 2, Development and Maintenance Environment. Acquires information about processing capabilities and development aids and tools used to develop, support, and maintain a computer program. This was intended to be completed by individuals such as project software managers with detailed knowledge of the software development and maintenance environment.

c. Part 3, Software Characteristics. Acquires information about the technical characteristics of a computer program, including design, code and data characteristics, and complexity. This part was intended to be completed by individuals such as lead programmers with detailed knowledge of the computer program.

d. Part 4, Software Tools. Acquires information about automated tools used for error detection and analysis and testing, including compiler-based tools and static and dynamic analyzers. This part was intended to be completed by individuals such as lead programmers or lead test engineers with general knowledge of the use of software tools. Ratings of tools used were intended to be done by tool users.

e. Part 5, Software Test Methods. Acquires information about testing activities for a computer program, including rationale for test cases and test completion criteria.

This part was intended to be completed by individuals such as software test engineers with indepth knowledge of software test activities and error reports.

f. Part 6, Software Error Categories. Acquires information about types of errors detected during initial development or maintenance. This part was intended to be completed by individuals such as software engineers, test engineers, configuration management personnel, or quality assurance personnel who are responsible for recording or maintaining software error data.

2.3 SURVEY FINDINGS

This section represents summaries of the characteristics and nature of software development, maintenance, and testing practices of each of the sites. The findings are derived from a composite of data collected from the questionnaire forms and by onsite interviews. Generally, the interviews were conducted with persons other than those who had completed the questionaires. Therefore, a different and broader perspective was obtained during the onsite interviews.

Since SAC did not choose to participate in the questionnaire phase of the survey but did participate in the onsite interviews, the level and character of data collected for this site differs from the other sites, although information was provided by participants during the visit to this site. This is particularly evident in the testing area, where sufficiently detailed data were lacking to make assessments about the general characteristics of the software testing environment at SAC. It should be noted that this is a more difficult site in which to characterize the testing environment because of the multimission nature of SAC responsibilities and the wide diversity of unrelated computing systems supported at this site.

2.3.1 Observations

The Air Force site summaries included in section 2.3.3 of this report provide objective data about those sites, to the extent that the survey participants were objective. In contrast to this information, certain subjective assessments about the nature of software environments at these sites were derived from the surveys. These assessments are included for the interest they may provide and are not necessarily substantiated by data.

a. The amount of the total project effort devoted to software development and maintenance requirements, implementation, testing, and installation was striking. Even in the hardware-oriented development programs, software development consumed a large share of the total effort. The impact of software processes on these systems is considerable and should not be underestimated.

b. There appeared to be relative uniformity among the Air Force sites in the application of software development methodologies. The sites surveyed generally employed system- and CPCI-level specifications, prepared them using MIL-STD-483 and -490, and used them for developing test requirements for system-level software testing. Also, software unit testing against design specifications was typically prepared according to MIL-STD-483 and -490. The sites made general use of design reviews (PDR and CDR) and adhered to Air Force regulations AFR 300-series and/or AFR 800-14. Recent technology for software development was used at the sites, including hierarchical design, program modularity, and incremental development techniques such as top-down or bottom-up integration. However, none of the sites were observed to be using the more recent, so-called advance techniques, such as formal specifications (InaJo, HDM), automated requirement languages (SREM/REVS), data- structured design techniques (Warnier, Orr, and Jackson), or formal testing techniques (symbolic execution). Of course, these omissions may be intentional, since these techniques do not lend themselves to application in these environments. Also, many of the surveyed computing systems have been in place for a number of years, preempting the application of such technology.

c. Both assembly language and compiler (HOL) debug aids are universally available; the compiler aids are widely applied and generally considered easy to use and effective. On the other hand, assembler debug aids seem to be somewhat avoided except where essential, probably due to their being more difficult to use.

d. Software testing aids are not widely available at the Air Force sites. The tools generally in use are directed at project test management and scheduling and at file and configuration management for testing purposes. Some specially developed tools were reported that aid in data manipulation and test data analysis, but they too were infrequent. The most frequently used tools that directly support testing activities are environmental and system simulators. They also are used to support

development activities.

e. The prevalent methods of testing are debug (compile and remove coding errors), functional testing to specification requirements, and system operation, "soak." Methodical approaches toward establishing test goals and objectives for particular circumstances and shaping the test cases to those goals and objectives were not evidenced by the survey. There probably are no uniform practices for defining test cases among the Air Force sites.

f. There was a uniformly high level of concern among the Air Force sites about the adequacy of their software testing programs. All the sites directed considerable attention to testing; and the testing programs conducted either by the Government or by contractors are structured, progressive, and subject to management visibility. In all cases, final approval of testing is required before the software is permitted to become operational.

g. Documentation uniformly prepared for software testing programs consists of test plans, test procedures, and test reports. V&V plans are not used at all sites.

h. Independent V&V is not a common practice among the sites surveyed; it appears to be applied to large, complex contractor development programs. No instances of its use were noted where the software is developed or maintained by the Air Force sites surveyed.

i. Structuring of test programs at the Air Force sites shows a common symmetry. Testing consisted of specific, well-defined series of phases: debug and unit/module testing, often defined as computer program test and evaluation (CPT&E); integration and CPCI verification, often identified as DT&E; system testing and operational verification, often referred to as OT&E.

j. Analytical, metrical, statement, or logic coverage methods for determining the completion of testing are seldom used at the Air Force sites surveyed. The three most commonly used completion criteria are specification coverage, satisfaction of test requirements, and schedule completion.

2.3.2 Air Force Site Summaries

This section of the report provides summaries of the Air Force sites surveyed.

Armament Division. This site is a developing agency for tactical weapon systems, particularly threat, missile, and scoring systems. All embedded software systems development at the Armament Division (AD) is performed by contractors. The contractors are usually small, specialized, high-technology companies, but larger aerospace companies also contribute to the systems development. The software contained in these systems typically tends not to be critical, even though the systems themselves may be critical. The contractors design, develop, and test the software according to contractor-defined standards, but under the general contractual-level supervision of Air Force personnel. Testing practices vary rather widely among the many contractors supporting the AD.

Aerospace Defense Center. The Aerospace Defense Center (ADC) is both a developing and user agency for a single mission area: strategic warning and support systems. However, its systems are currently in place and undergoing maintenance activities, which consist of performance enhancements and additional system capabilities. Development is accomplished extensively by contractor personnel, depending on the organization and system function. Maintenance is either conducted by the Air Force or by contractors under close Air Force supervision. The system components are highly interrelated and the functions they perform tend to be highly critical. Systems are typically redundant, with multilevel fallback considerations. The computational systems perform numerous real-time communication processing functions, including message switching and routing control, complex trajectory calculations, systems status monitoring, and man-machine interface for control purposes. Testing practices are relatively uniform among the functional software activities and are highly adapted to the system characteristics. The space activity employs an independent test approach to interface and system-level testing after the programmer has completed module testing. Detailed test procedures are developed and updated prior to system testing. The requirement for successful software testing at both the module and system level is deeply embedded within the version release cycles. Operational testing to mission specifications is accomplished by the user subsequent to turnover from the software organization.

Aeronautical Systems Division. Aeronautical Systems Division (ASD) is a developing agency for weapon systems equipment, including avionics, automatic test equipment, crew training devices, flight control and reconnaissance, and C^3 systems. System and software development are typically contracted. The development contractors tend to be medium- to large-size aerospace corporations, with substantial technical expertise in weapon systems development. The systems and embedded software are developed under well-defined contractual requirements and monitored by onsite representatives with frequent reviews of activities and documentation by ASD personnel. A wide diversity of software is developed by ASD, including numerous aircraft avionics and control systems, and communications systems software. Development activities are controlled by Government standards, and testing practices are fairly uniform, adhering to Air Force regulations and uniformly defined testing requirements imposed by SOW.

Strategic Air Command. The Strategic Air Command (SAC) has a diversity of missions to support (e.g., C^3, war planning, intelligence support, and strategic weapons support) and develops a wide diversity of unrelated systems for these missions. For strategic weaponry, SAC is a user agency, while for the other areas it is both a developer and user. War planning and intelligence systems are developed and maintained almost exclusively by Air Force personnel, while the development of information and management systems often is conducted primarily by contractors and the maintenance shared by Air Force and contractor personnel. The software developed for the warning functions ranges from highly critical to noncritical. Software development practices for contractors are controlled by the SOW, and internal maintenance is conducted in accordance with SAC regulations. The computation systems used by SAC tend to be data-base and data-processing intensive, such as in the intelligence and war planning areas. The warning area includes real-time control functions, and the command centers use C^3 technology software. SAC-conducted software testing practices and methods are standardized by SAC regulations. However, there exists variability in their application, corresponding to the differences in the software categories, criticality, and functional organizational practices.

Space Division. The Space Division (SD) is a development agency for space-related systems, including satellites, launch vehicles, and ground control and communications systems. SD relies extensively on contractors to develop its systems and the embedded software. These contractors also perform maintenance under follow-on contracts.

Software development requirements are defined in detail in the SOW; SD personnel, often coupled with technical consultant contractors, intensively monitor all development activities at all levels. Frequent reviews and technical direction are provided by this agency. A wide diversity of software categories is developed by SD, including software for communications, satellite control systems, prelaunch checkout and ground test systems, space vehicle avionics and control, and system simulations. This site employs independent verification and validation (IV&V) contractors to a greater extent than any of the other sites surveyed. Software testing practices are established by Air Force regulation and defined by the SOW. As a result, these practices tend to be relatively uniform among the development contractors. SD places great emphasis on the thoroughness, sufficiency, and formality of contractor testing practices.

Tactical Air Command. The Tactical Air Command (TAC) is the development and user agency for the major Air Force tactical planning system, the Computer-Assisted Air Force Management System (CAFMS). The CAFMS is a single-function, highly interrelated automated processing system. The major output product of CAFMS is the air tasking order report. CAFMS was developed by TAC personnel with some contractor assistance during the early requirement and design phases. Management, development, and maintenance of this system are well defined and uniquely adapted for its ongoing support. The system is currently operational, but undergoes continual enhancements and incorporation of new capabilities. The overall function of the CAFMS is quite critical, but few of its software components are considered to be more than moderately critical. The system does incorporate some automated fallback provisions in case of failure, but redundancy of system functions is not provided, and reversion to manual operation is the ultimate fallback provision. Testing practices are well defined and are incorporated as an integral part of a version release management system developed by TAC specifically for CAFMS. Testing is applied uniformly to all software components undergoing development.

3.0 SUMMARY OF TASK 2

3.1 EVALUATION OF TESTING TECHNIQUES

The relationship between mission application test requirements and state-of-the-art software test techniques was developed in this task. A step-by-step approach was applied in developing the tables used in the guidebook. This process began with developing the taxonomies to be used in developing the guidebook. First, the software error categorization was selected; second, the method was selected for organizing the types of software in the five USAF missions areas.

The first table constructed was the testing confidence level table. This table allows the guidebook user to establish a confidence level of software testing appropriate to the criticality of the software, the type of software, the budget and schedule constraints, etc. The testing confidence level number is then used in subsequent tables in the guidebook to extract appropriate testing tools and techniques for a specific situation.

The next table constructed rated the effectiveness of software test techniques for detecting specific software error types. This table is one of the few that does not use the testing confidence level. It uses an independent rating system that considers only the relative effectiveness of the specific tool or techniques being considered against specific software error types. This was followed by the construction of a table that related the tools and techniques to specific software types in terms of the testing confidence level. The software categories were derived from a survey of Air Force software testing practice. The relative criticality of the software, and its difficulty from a software engineering viewpoint were used to define the 18 categories used in the guidebook. These categories are listed in section 3.4 of this final report, and defined in table 2.2-2 of the guidebook.

A table that was to relate specific software error types to Air Force missions did not prove to be feasible. The task 1 survey of Air Force sites revealed that there was not enough data on software errors to construct the table. An attempt to provide a generic table was dropped when it became evident that such a table was too general; it contained no new or useful information.

A table that was to relate test phases to test techniques was reconsidered for its effectiveness. In some cases, the guidebook user may know his testing objectives, but in other cases the user may only know the relevant test phase. Therefore, this table was redesigned to relate test objectives (a more basic concern) to relevant test techniques. A new table was designed to relate test phases to test objectives. This change results in a more accessible and usable methodology for guidebook users, with varying kinds and levels of knowledge of their software environment and testing problems. A basic premise of this table, resulting from both state-of-the-art theory and discussions with Air Force test personnel, was that testing techniques used in early test phases remain applicable in later phases. For example, a code auditor, used in unit and module testing, should be used in later phases to assure that modifications and corrections resulting from later test phases do not violate coding conventions.

As a result of the discussion of the draft technical report on task 2 at an oral technical review, the last two tables were combined into one large table. The attendees agreed that the new combined table included in the guidebook was a more efficient and usable design than the two-table concept.

A table that relates software test techniques to support tools was constructed in a similar manner to the other tables. The major difference is that the table does not use the testing confidence level values as entries in the matrix. Instead, the support tools are merely indicated as being appropriate or inappropriate for the various test tools.

3.2 TASK 2 CONSIDERATIONS

Many considerations were involved in developing the contents of the tables. The ratings given in each entry reflect testing confidence level in terms of the complexity of the software, criticality of the software to the mission, and relevance of each software test tool or test technique to the particular software type.

The software categories were also chosen carefully so that the mission software types could be mapped into the software categories used in the guidebook. Within a category, software is chosen for similarity of function, internal structure, and complexity.

Two approaches were used to accomplish the complex task of evaluating the various testing techniques. First, a comprehensive literature search provided (1) a source of information on the various types of test techniques available and (2) a first cut at rating their effectiveness. Details on classes of testing techniques were gathered from articles on individual techniques, and relative ratings of technique effectiveness and limitations were gathered from survey articles. Second, a group of Boeing software engineers who had extensive experience in tools and testing were surveyed. This survey asked for the techniques to be rated according to each of the 10 considerations described in the following paragraphs.

a. Performance analysis capability. Determined by literature search, personal experience, and discussions with other software engineers.

b. Error detection and location capability. We rated each method, efficiency, relative success at detection of specific error types, and the precision with which the techniques locate (i.e., precisely identify the exact error) the software errors so that they can be understood, analyzed, and corrected efficiently.

c. Side effects and benefits. Other worthwhile results of the technique were also considered. For example, some techniques may provide output that can be used in the product documentation of the software being tested.

d. Cost and schedule benefits and impact. Specific techniques differ widely in their relative costs, the time involved in their use (schedule impact), and their relative benefit.

e. Management impact. All techniques were rated as to their benefit to management. Some approaches provide valuable visibility on the progress and health of the software development project. If a technique provided additional visibility to management, it was rated more valuable and productive (given a confidence level rating that would make its use more likely). It should be noted that some techniques such as a peer code review specifically exclude management visibility to achieve their goals.

f. Training required. Some techniques may be esoteric and require extensive training or personnel with special skills or backgrounds (e.g., algebraic and symbolic analysis). Other techniques involve common skills and are simple to apply (e.g., design and code walkthroughs).

g. User constraints. General rather than specific user constraints were considered in evaluating the techniques. The basic features characteristic of software test tools and techniques were used to determine the constraints imposed on the user. A tool may be available now only on the computers of a specific vendor, which limits the usefulness of that tool. However, in 6 months or a year, a similar tool may be available on many machines. In building rating tables for the guidebook, we considered the generic features, not the specific software test tools. The guidebook will provide considerations for evaluating specific tools and techniques.

h. Typical computer resources required. Each technique was evaluated in terms of its computer resources (time and storage) requirements compared to the potential benefits. Some techniques, such as peer code review, require little or no computer resources; whereas, other approaches (random testing or real-time testing) may use large blocks of time, large blocks of primary and secondary storage, or a considerable number of other computer resources.

i. Level of human interaction required. The various techniques considered in the tables vary considerably in the level of human interaction required. This human interaction must be considered at several levels: first, the amount of time the human must be involved compared to potential benefits; second, the degree of expertise required to effectively use the technique. The greater the proportional amount of time required or the level of expertise, the less attractive the technique was rated.

j. Usefulness of techniques in supporting modern programming practices. If the techniques encouraged the use of modern programming practices by the developers, or produced results useful for modern programming practices, they were rated more attractive.

3.3 SELECTION OF TOOL TAXONOMY

An understandable, comprehensive, user-friendly taxonomy of software test techniques was necessary for building the tables. Several taxonomies were evaluated as to their effectiveness according to the following criteria: they should be easy to understand, rational and systematic, free of inconsistencies, compatible with the intent of the guidebook, recognizable by the target audience, and compatible with the test phase relationship. The taxonomy has four major categories:

- Static analysis.
- Dynamic analysis.
- Symbolic testing.
- Formal analysis.

The first two categories include many software testing techniques. The complete taxonomy is described in detail in the following paragraphs.

Static Analysis. This is an automated analysis of computer program source code without executing the computer program. The major subcategories of static analysis are—

a. Code reviews and walkthroughs.
   1. Peer Review - review of code and design by project personnel.
   2. Formal Review - review by customer at scheduled points in the life cycle.

b. Error and anomaly detection techniques:
   1. Code Auditing - automated review of source code with respect to prescribed programming standards.
   2. Interface Checking - analysis of interface for consistency and completeness.
   3. Physical Units Checking - analysis of units of measure for consistency.
   4. Data Flow Analysis - analysis of sequences of program events to locate errors.

c. Structure analysis techniques and documentation:
   1. Structure Analysis - analysis of design or program structure to identify logical flaws.
   2. Documentation - production of documentation resulting from analysis (e.g., set-use listings).

d. Program quality analysis:

1. Halstead's Software Science - an attempt to formulate fundamental relationships for all computing languages.

2. McCabe's Cyclomatic Number - an assessment of program complexity based on the number of branches.

3. Software Quality Measures - a system to predict various software qualities (e.g., reliability, maintainability) based on a multitude of small, discrete measures, called metrics.

e. Input space partitioning. A path in a program consists of a possible flow of control. In path analysis, the input space is partitioned into path domains—those subsets of the program input domain that cause execution of the paths.

1. Path analysis - detection of missing paths or incorrect paths.
2. Domain testing - selection of test data on or near domain boundaries.
3. Partition analysis - a method in which specifications of the program are partitioned into subspecifications that are then matched with domain partitions to generate a more sensitive test.

f. Data-flow guided testing. This is a method for obtaining structural information about programs (widely used for compiler design and optimization). One result is a set of dynamically meaningful relationships among program variables. Control flow information about the program is then used to construct test sets for the paths to be tested.

Dynamic Analysis. This is a method of analyzing a computer program in which the program itself is executed on a computer. The major subcategories of dynamic analysis are—

a. Instrumentation-based testing. Programs are instrumented by statements or routines that do not affect the functional behavior of the program but record properties of the executing program.

1. Path and structural analysis techniques - analysis of test coverage, execution frequency of branches, statements, etc.

2. Performance measurement techniques:
   (a) Timing and resource analysis - analysis of execution time and computer storage resources required by the software.
   (b) Algorithm complexity analysis - analysis of algorithm using a formalized approach.

3. Assertion checking - a technique using assertion statements in the source code that are then checked for validity during execution.

4. Debug aids - various facilities permitting trace, breakpoint, register inspection during or following execution.

b. Random testing. This is a black-box technique in which a program is tested by randomly sampling inputs.

c. Functional software testing. The specification of the program is viewed as an abstract description of its design and is then used as a guide to generate functional test data. Extremal and special values are the most important values in the input domain of a variable.

d. Mutation testing. Mutation testing is a technique for evaluating test data adequacy. The program under test is changed (forming mutants of the original); test data are applied to the mutants. If the test data uncover the mutants, the data are accomplishing its job; if not, either the program is still correct in its mutated form, or the test data were inadequate to locate the mutant error.

e. Real-time testing. This is the testing of software on "host" computers using environment simulators, as well as the testing of software on the "target" computer in the actual hardware or software system, or a simulation thereof.

Symbolic Testing. The input data and variables of a program are given formal or symbolic values, and the possible executions are characterized formally. The execution of the program is simulated by a symbolic evaluator that interprets the formal representation of the program and data.

Formal Analysis. This is a formal method of proving a design correct and uses rigorous mathematical techniques to analyze the algorithms of a solution. At present, formal analysis is primarily a manual activity with limited automated assistance.

This taxonomy provided the most useful working categorization for test techniques for the internal tables of the Software Test Guidebook.

3.4 SOFTWARE ENVIRONMENTS CHARACTERISTIC OF USAF MISSIONS

The task 1 survey provided information on the kinds of software characteristics of each USAF mission. A first draft of a table listing all the software types characteristic of each mission area was prepared, based on the task 1 interim technical report. This draft was submitted to the mission focal points for their review. The comments and criticisms were then used to build a final list.

There was much overlap between the mission areas, and a method of efficiently characterizing them for the guidebook was needed. Several software classification schemes were produced and evaluated. Guidelines used for evaluating classification schemes were the same as those used in selecting a tool taxonomy: the structure should be parallel in nature, the software types should be as unique as possible, and the terminology should be as clear as possible. A software classification scheme was chosen, and all the software types characteristic of the USAF missions were classified using that scheme. The classification of software categories follow.

- Batch (general).
- Event control.
- Process control.
- Procedure control.
- Navigation.
- Flight dynamics.
- Orbital dynamics.
- Message processing.
- Diagnostic software.
- Sensor and signal processing.
- Simulation.

- Database management.
- Data acquisition.
- Data presentation.
- Decision and planning aids.
- Pattern and image processing.
- Computer system software.
- Software development tools.

4.0 SUMMARY OF TASK 3

4.1 GUIDEBOOK DEVELOPMENT

Task 3 comprised two parts: first, design and prepare the guidebook; second, write a final report to describe the entire three-task research effort. The first part comprised about 95% of the task 3 effort and will be the only subject of this section.

The design of the guidebook was largely a matter of building a text structure to support the revised table structure from task 2. The guidebook preparation constituted two main efforts. The first effort was to prepare the text. One part of the text preparation was to write the directions and guidelines for tool selection. The second part of the text preparation was to assemble material that described all testing techniques and methodologies. The second effort was to prepare graphic material for the guidebook. This included the job of (1) rebuilding the guidebook tables to conform to the Air Force direction given at the task 2 oral report and (2) preparing graphic material to explain guidebook usage.

4.2 PREPARATION OF TEXT

The guidebook design was based on the table design completed in tasks 1 and 2. Several changes were made to the outline to structure the guidebook so that it is easy to use.

Directions and guidelines were written so that they are compact, clear, and easy to use. Thus, these sections are short and contain only general guidelines and considerations. Complete descriptions of techniques and their implementation were put into a separate section. This design frees the instruction sections of technique-specific information to make the methods of technique selection more obvious.

Most of the material for the section on technique descriptions was derived from two National Bureau of Standards publications: NBS Special Publications 500-93, "Software Validation, Verification, and Testing Techniques and Tool Reference Guide," (POW82A) and 500-98, "Planning for Software Validation, Verification, and Testing," (POW82B). This material was reviewed and updated as necessary.

4.3 PREPARATION OF GRAPHIC MATERIALS

As a result of the task 2 oral review, several major changes were made in the tables used for test technique selection. The basic categorization of testing techniques (taxonomy) was restructured. In addition, two of the tables in the original design were combined into one table.

The structured tables were compact and efficient to use, but difficult to understand at first glance. To correct this situation, a number of new diagrams were designed and built on the word processor. The diagrams explained in a step-by-step graphic manner how the guidebook and tables were to be used.

4.4 EDITING AND REVIEW

Technical editing was done first by the authors, who checked one anothers material. The guidebook was then reviewed by the project manager. After completing the first draft, an independent technical editor was assigned to review the guidebook.

5.0 BIBLIOGRAPHY

(ADA79) "Ada Environment Workshop." Sponsored by DoD High Order Language Working Group, November 27-29, 1979, San Diego.

(ADR81) Adrion, W. R., M. A. Branstad, and J. C. Cherniavsky. "Validation, Verification and Testing of Computer Software." NBS Special Publication 500-75, Superintendent of Documents, U.S. Documents, U.S. Govt. Printing Office, Wash, D.C., February 1981.

(AHO74) Aho, A.V., J.E. Hopcroft, and J.D. Ullman. The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading, Mass., 1974.

(AHO77) Aho, A.V. and J.D. Ullman. Principles of Compiler Design. Addison-Wesley, 1977.

(AIR78) "Airborne Systems Software Acquisition Engineering Guidebook for Verification, Validation, and Certification", Air Force document, ASD-TR-79-5028, 1978.

(ALF76) Alford, M. W. "A Requirements Engineering Methodology for Real-Time Processing Requirements," TRW Software Series, TRW-SS-76-07, Systems Engineering and Integration Division, Sept., 1976.

(ALF79) Alford, M. A. "Theoretical Foundations of Toolsmithing." Proceedings of COMPSAC 79, IEEE Catalog No. 79CH1515-6C, November 1979.

(ALL76) Allen, F.E. and J. Cocke. "A Program Data Flow Analysis Procedure", CACM, Vol. 19, No. 3, March 1976.

(ARE80) "A Review of Software Maintenance Technology." (RADC TR-80-13) dtd Feb 80. Available from the U.S. Dept. of Commerce, National Technical Information service, 5285 Port Royal Road, Springfield, Virginia, 22151, Accession No. A083-985.

(AUT77) "Automated Data Systems Documentation Standards." Department of Defense standard 7935.1-S, September 1977.

(AUT79) "Automated Software Tools Catalog." Boeing Computer Services document 10236, 1979.

(BAR78) Barbuto, P., Jr. and J. Geller. "Tools for Top-Down Testing." Datamation, Vol. 24, No. 10, October 1978, pp. 178-182.

(BAR80) Barry, M. "Airborne Systems Software Acquisition Engineering Guidebook for Software Testing and Evaluation." TRW report ASD-TR-80-5023, March 1980.

(BEL74) Bell, D.E. and J.E. Sullivan. "Further Investigations into the Complexity of Software". (Tech. Rep. MTR-2874). Bedford, MA: MITRE. 1974.

(BEN78) Benson, J.P. and S.H. Saib. "A Software Quality Assurance Experiment", Proceedings of the Software Quality and Assurance Workshop, San Diego, Nov. 1978.

(BEN79) Bentley, J.L. "An Introduction to Algorithm Design", Computer, Feb. 1979.

(BLA71) Blair, J. "Extendable Non-Interactive Debugging." Debugging Techniques In Large Systems, Prentice-Hall, Englewood Cliffs, N.J., 1971, pp. 93-115.

(BOE75) Boehm, B., R. McClean, and D. Urfrig. "Some Experience with Automated Aides to the Design of Large-scale Reliable Software", IEEE Transactions of Software Engineering, SE-1, 1975(125-133).

(BOY75) Boyer, R.S., B. Elspas, and K.N. Levitt. "SELECT-A Formal System for Testing and Debugging Programs by Symbolic Execution", Proceedings of the International Conference on Reliability of Software, April 1975.

(BRA75) Bratman, H. and T. Court. "The Software Factory." Computer, May 1975, pp. 28-37.

(BRA77) Bratman, H. and M. C. Finfer. "Software Acquisition Management Guidebook: Verification." System Development Corporation report ESD-TR-77-263, August 1977.

(BRO75) Brooks. The Mythical Man-Month. Addison-Wesley, 1975.

(BRO77) Brown. "Impact of Modern Programming Practices on System Development." RADC-TR-77-121, 1977.

(BRO78) Brown, J.R. and K. Fischer. "A Graph Theoretic Approach to the Verification of Program Structures", Proceedings of the 3rd International Conference on Software Engineering, May 1978.

(BRO82) Brownell, L. "Jovial J73 Code Auditor", Technical Report, Proprietory Software Systems, Inc., 9911 W. Pico Blvd., Los Angeles, CA 90035, 16 March 1982.

(BRO83) Brown, P.J. "Error Messages: The Neglected Area of Man/Machine Interface?", CACM, Vol. 26, No.4, April 1983.

(BUL74) Bulut, N. and M.H. Halstead. "Impurities Found in Algorithm Implementation", SIGPLAN Notices, 1974.

(BUX80) Buxton J. N. "An Informal Bibliography on Programming Support Environments." SIGPLAN Notices, December 1980.

(CAM81) Campbell, O. and S. Saib. "Embedded Software Verification Through Instrumentation", NAECON 81, May 19-21, 1981, Vol. I, pp. 395-401.

(CHE79) Cheatham, T.E., G.H. Holloway, and J.A. Townley. "Symbolic Evaluation and the Analysis of Programs", IEEE Transaction on Software Engineering, SE-5,4, July 1979.

(CHO) Chow, T.S. "A Generalized Assertion Language", Proceedings 2nd ICSE, S.F., Calif., pp. 392-399.

(CLA76) Clarke, L.A. "A System to Generate Test Data and Symbolically Execute Programs", IEEE Transaction of Software Engineering, SE-2, Sept. 1976.

(CLA78) Clarke L. "Top-Down Testing with Symbolic Execution." Digest for the Workshop on Software Testing and Test Documentation, Ft. Lauderdale, Florida, 18-20 December 1978, pp. 191-196.

(COC70) Cocke, J. and T.J. Schwartz. Programming Languages and Their Compilers, Preliminary Notes, Second Revised Version, Courant Institute of Mathematical Sciences, New York, 1970.

(COD76) "Code Reading Structured Walk-Throughs and Inspections", IBM, IPTO, Support Group, World Trade System Center, Postbos 60, Z- -nmeer, Netherlands, March 1976.

(CON70) "Control Flow Analysis". SIGPLAN Notices, 1970, pp. 1-19.

(COR76) Cornall, L.M. and M.H. Halstead. "Predicting the Number of Bugs Expected In a Program Module", (Tech. Rep. CSD-TR-205). West Lafayette, IN: Purdue University, Computer Science Department, October 1976.

(CRO75) Crocker, S. and B. Balzer. "The National Software Works: A New Distribution System for Software Development Tools." Workshop on Currently Available Testing Tools, April 1975, p. 21.

(CUR79) Curtis, B., S.B. Sheppard, and P. Milliman. "Third Time Charm: Stronger Prediction of Programmer Performance by Software Complexity Metrics", Proceedings of the Fourth International Conference on Software Engineering. New York: IEEE, 1979.

(DAL77) Daly, E.B. "Management of Software Development", IEEE Transactions on Software Engineering, May 1977.

(DAV) Davis, C.G. "Testing Large, Real-Time Software Systems", Infotech State-of-the-Art Report - Software Testing, Infotech International, Berkshire, England, Vol. 2, pp. 85 - 105.

(DEC78) DEC IAS/RSX-11 "Utilities Procedure Manual", Digital Equipment Corporation, 1978.

(DEF79) "Defense Mapping Agency: Modern Programming Environment Study-Final Technical Report." Boeing Computer Services and Planning Systems International, Contract No. SB 1438(A)-79-C-001, October 1979.

(DEM79) DeMillo, R.A., R.J. Lipton, and F.G. Sayward. "Program Mutation: A New Approach to Program Testing", Infotech State-of-the-Art Report on Software Testing, V.2, INFOTECH/SRA, 1979, pp.107-127.

(DEM83) DeMillo, R. A., and R. J. Martin. "The Software Test and Evaluation Project: A Progress Report." Proceedings of the National Conference on Software Test and Evaluation, 1-3 February 1983.

(DER76) DeRemer and Kron. "Programming-in-the-Large Versus Programming-in-the-Small." IEEE Transactions on Software Engineering, June 1976.

(DEU81) Deutsch, M.S. "Software Project Verification and Validation", IEEE Computer, April 1981.

(DON80) Donahoo, J. D. and D. Swearinger. "A Review of Software Maintenance Technology." RADC report RADC-TR-80-13, February 1980.

(ELM73) Elmendorf, W.R. "Cause-Effect Graphs in Functional Testing", TR-00.2487, IBM Systems Development Division, Poughkeepsie, New York, 1973.

(ELS76) Elshoff, J.L. "Measuring Commercial PL/1 Programs Using Halstead's Criteria", SIGPLAN Notices, 1976,11, 38-46.

(ELS72) Elspas, B., et. al. "An Assessment of Techniques for Proving Program Correctness", ACM Computing Surveys, 4, June 1972.

(END75) Endres. "An Analysis of Errors and Their Causes in System Programs." IEEE Transactions on Software Engineering, June 1975.

(FAC77) "Factors in Software Quality", Final report, RADC-TR-77-369, 1977.

(FAG76) Fagan, M.E. "Design and Code Inspections to Reduce Errors in Program Development", IBM Systems Journal, No. 3, 1976.

(FAI78) Fairley, R. E. "Tutorial: Static Analysis and Dynamic Testing of Computer Software." Computer, April 1978, pp. 14-24.

(FEL79) Feldman. "MAKE - A Program for Maintaining Computer Programs." Software Practice and Experience, April 1979.

(FIS74) Fischer, K.F. "User's Manual for Code Auditor, Code Optimizer Advisor, Unit Consistency Analyses", TRW Systems Group, Redondo Beach, Calif., July 1974.

(FIT78) Fitzsimmons, A.B. and L.T. Love. "A Review and Evaluation Science". ACM Computing Surveys, 1978, 10, 13-18.

(FLE79) Fleiss, J., G. Phillips, and A. Alvarez. "Compiler Acceptance Guidebook." RADC report RADC-TR-77-148, May 1979.

(FLO67) Floyd, R.W., T.J. Schwartz, editor. "Assigning Meanings to Programs", Mathematical Aspects of Computer Science, 19, American Mathematical Society, Providence, R.I., 1967.

(FOS76) Fosdick, L.D. and L.J. Osterweil. "Data Flow Analysis in Software Reliability", ACM Computing Surveys, 8, pp.305-330, Sept. 1976.

(FRE77) Freedman, D.P. and G.M. Weinberg. Ethno-Technical Review Handbook. Ethnotech, Inc., 1977.

(FUM76) Fumani, Y. and M.H. Halstead. "A Software Physics Analysis of Akiyama's Debugging Data", Proceedings of the MRI 24th International Software Engineering. New York: Polytechnic Press, 1976.

(GAN79) Gannon, C., R.N. Meeson, and N.B. Brooks. "An Experimental Evaluation of Software Testing - Final Report," General Research Corp., CR-1-854, sponsored by Air Force Office of Scientific Research, May 1979.

(GAN80) Gannon, C. "Jovial J73 Automated Verification System-Study Phase", RADC-TR-80-261, August 1980, NTIS accession no. A091-190.

(GIL77) T. Gilb, "Software Metrics," Winthrop Publishers, Inc., Cambridge, Mass, 1977.

(GLA76) Glass, R. L. "An Experiment in the Use of Analyzers as a Computer Software Reliability Tool in the BAC Project Environment", Boeing Aerospace Company D180-19987-1, August 1976.

(GLA78) Glass, R. L. "Software Reliability Methodology Survey and Guidebook", The Boeing Company, D180-22930-1, 1978.

(GLA79A) Glass, R. L. Software Reliability Guidebook. Prentice-Hall, 1979.

(GLA79B) Glass, R. L. "Software Reliability at Boeing Aerospace: Some New Findings." *Boeing Aerospace Co.* report D180-25392-1, September 1979.

(GLA79C) Glass, R. L. "Real Time Software Debugging and Testing: Proposed Solutions." Boeing Aerospace Co. report D180-25249-3, September 1979.

(GLA81) Glass, R. L. and R. Noiseux. Software Maintenance Guidebook. Prentice Hall, Inc., 1981.

(GLA-A) Glass, R. L. "Automated Tools for Software IV & V." The Boeing Co. Unpublished draft.

(GLA-B) Glass, R. L. "Recommended: A Minimum Standard Software Toolset". The Boeing Co., Unpublished draft.

(GOD77) Godoy and Engels. "Software Sneak Analysis." Proceedings of the AIAA Conference on Computers in Aerospace, 1977.

(GOE81) Goel, Dr. Amrit. "A Guidebook for Software Reliability Assessment", pr under contract No. F30602-81-C-0169, for RADC.

(GOO75) Good, D.I., R.L. London, and W.W. Bledsoe. "An Interactive Pı Verification System", Proceedings of the 1975 International Confere Reliable Software, IEEE Catalog #75CH0940-7CSR.

(GOR76) Gordon, R.D. and M.H. Halstead. "An Experiment Comparing FOI Programming Times with the Software Physics Hypothesis", AFIPS Pr ings, 1076, 45, 935-937.

(HAL73) Halstead, M.H. "An Experimental Determination of the 'Purity' of a Algorithm", ACM SIGME Performance Evaluation Review, 1973, 2(1), 1

(HAM82) Hamer, P.G., G.D. Frewin. "M. H. Halstead's Software Science-A (Examination", Proceedings of the 6th International Conference on Sc Engineering, Tokyo, Japan, September, 1982.

(HAR71) Hartwic, R.D. "The Advanced Targeting Study", SAMSO-TR-71-124. June 1971.

(HEC73) Hecht, M.S. and J.D. Ullman. "Analysis for a simple algorithm for glot flow problems". Proc. ACM SIGACT/SIGPLAN Symp. on Princi Programming Languages. Boston, Mass., Oct. 1973.

(HEC75) Hecht, M.S. and J.D. Ullman. "A Simple Algorithm for Global Dat Analysis Problems", Siam Journal of Computing, Vol. 4, No. 4, Decembe

(HEC81A) Hecht, H. "Synopsis of Interviews from a Survey of Software A Tool SoHaR Inc. report NBSIR 81-2388, November 1981.

(HEC81B) Hecht, H. "Final Report: A Survey of Tool Usage." NBS Special Pub 500-82, Superintendent of Documents, U.S. Govt. Printing Office, Was 20402, November 1981.

(HEC82) Hecht, H. "The Introduction of Software Tools." NBS Special Publication 500-91, Superintendent of Documents, U.S. Govt. Printing Office, Wash, D.C. 20402, September 1982.

(HEI82) Heidler, W., et al. "Software Testing Measures." RADC-TR-82-135, May 1982.

(HET73) Hetzel, W. Program Test Methods. Prentice-Hall, 1973.

(HET76) Hetzel, W. "An Experimental Analysis of Program Verification Methods", Ph.D. Thesis, University of North Carolina, 1976.

(HOA61) Hoare, C.A.R. "Partition (Algorithm 63) and QUICKSORT (Algorithm 64)", CACM, Vol.4, No. 7, July 1961.

(HOA64) Hoare, C.A.R. "QUICKSORT", Computer Journal, Vol.5, No.1, 1964.

(HOA71) Hoare, C.A.R. "Proof of a Program: FIND", CACM, Vol.14,No.1, Jan. 1971, pp.39-45.

(HOR75) Horowitz. Practical Strategies for Developing Large Software Systems. Addison-Wesley, 1975.

(HOR78) Horowitz, E. and S. Sahni. Fundamentals of Computer Algorithms. Computer Science Press, Potamac, MD. 1978.

(HOU81A) Houghton, R. C., Jr. "Features of Software Development Tools." NBS Special Publication 500-74, Superintendent of Documents, U.S. Govt. Printing Office, Wash, D.C. 20402, February 1981.

(HOU81B) Houghton, R. C., Jr., editor. "Proceedings of NBS/IEEE/ACM Software Tool Fair." NBS Special Publication 500-80, Superintendent of Documents, U.S. Govt. Printing Office, Wash, D.C. 20402, October 1981.

(HOU82) Houghton, R. C., Jr. "Software Development Tools." NBS Special Publication 500-88, Superintendent of Documents, U.S. Govt. Printing Office, Wash, D.C. 20402, March 1982.

(HOW A) Howden, W. E. "Functional Testing and Design Abstractions". A Journal of Systems and Software (to appear).

(HOW) Howden, W. E. "Symbolic Testing - Design Techniques, Costs, and Effectiveness", U.S. Dept. of Commerce, NTIS PB-268, 517, Springfield, VA.

(HOW75) Howden, W. E. "Methodology for Generation of Program Test Data", IEEE Transactions on Computers, TC-24, May,1975.

(HOW77) Howden, W.E. "Symbolic Testing and the Dissect Symbolic Evaluation System", IEEE Trans. on Software Engineering, Vol. SE-3, No. 4, July 1977.

(HOW78A) Howden, W. E. "An Evaluation of the Effectiveness of Symbolic Testing." Software Practice and Experience, July 1978.

(HOW78B) Howden, W. E. "Selection of Fortran Static Analysis Techniques." University of Victoria report DM-147-IR, August 1978.

(HOW78C) Howden, W. E. "Functional Program Testing." University of Victoria report DM-146-IR, August 1978.

(HOW79) Howden, W. E. "An Analysis of Software Validation Techniques for Scientific Programs." University of Victoria report DM-171-IR, March 1979.

(HOW80A) Howden, W. E. "Validation of Scientific Programs", U.S. National Bureau of Standards, Wash., D.C., 1980.

(HOW80B) Howden, W. E. "Completeness Criteria for Testing Elementary Program Functions", University of Victoria, Department of Mathematics, May 1980.

(HOW80C) Howden, W. E. "Functional Program Testing". IEEE Transactions on Software Engineering, SE-7, March 1980.

(HOW80D) Howden, W. E. "Functional Testing and Design Abstractions". Journal of Systems and Software, Vol. 1, 307-313, 1980.

(HUA75) Huang. "An Approach to Program Testing", ACM Computing Surveys, September 1975.

(HUA79) Huang, J.C. "Detection of Data Flow Anomaly Through Program Instrumentation", IEEE Trans. on Software Engineering, Vol. SE-5, No. 3, May 1979.

(JAC71) Jackson and Bravdica. "Software Validation of the Titan IIIC Digital Flight Control System Using a Hybrid Computer." Proceedings of the 1971 Fall Joint Computer Conference.

(KAR78) Karr, M., D.B. Loveman, III. "Incorporation of Units into Programming Languages", CACM, Vol.21, No.5, pp.385-391, May 1978.

(KEN76) Kennedy, Ken. "A Comparison of Two Algorithms for Global Data Flow Analysis", Siam Journal of Computing, Vol. 5, No. 1, March 1976.

(KER76) Kernighan B. W. and P. J. Plauger. Software Tools. Addison-Wesley, 1976.

(KIL73) Kildall, G.A. "A unified approach to global program optimization". Proc. ACM SIGACT/SIGPLAN Symp. on Principles of Programming Languages. Boston, Mass., Oct. 1973.

(KIN76) King, J.C. "Symbolic Execution and Program Testing", CACM, 19, 7, July 1976, pp.385-394.

(KIN81) King, Bill. "ARGUS on BITS." Boeing Computer Services-SAMA Software Engineering Technology, November 1981.

(LAP74) L. J. LaPadula, "Engineering of Quality Software Systems, Volume VIII, Software Reliability Modeling and Measurement Techniques," RADC-TR-74-325, Mitre Corp., Bedford, Mass., 1975 (NTIS AS/A-007773).

(LIP78) Lipton, R.J. and F.G. Sayward. "The Status of Research on Program Mutation", Digest of the Workshop on Software Testing and Test Documentation, Fort Lauderdale, FLA., 1978, pp.355-373.

(LOV76) Love, L.T. and A. Bowman. "An Independent Test of the Theory of Software Physics", SIGPLAN Notices, 1976,11 pp.42-49.

(MAN74) Mangold, E.R. "Software Error Analysis and Software Policy Implications", IEEE EASCON, 1974, pp. 123-127.

(MAN78) Manna, Z. and R. Walding. "The Logic of Computer Programming", IEEE-TSE, SE-4, No.3, May 1978, pp.199-229 (especially pages 199-204).

(MAR78) "Problem Program Evaluator (PPE) User Guide", Boole and Baggage, Inc., Sunnyvale, Calif., March 1978.

(MCC76) McCabe, T.J. "A Complexity Measure", IEEE Transaction on Software Engineering, Vol. SE-2, No.4, December 1976.

(MEL79) Melton, R. "Fortran Automated Verification System (FAVS)", Vol. I, technica report, RADC-TR-78-268, Jan. 1979, NTIS accession no. A065-405.

(MEL81) Melton, R. "Cobol Automated Verification System - Study Phase", RADC-TR 81-11, March 1981, NTIS accession no. A098-755.

(MER81) Merilatt, R. L., M. K. Smith, and L. L. Tripp. "Computer Software Verifica tion and Validation: A General Guideline." Boeing Computer Services repo BCS-40342, June 1981.

(MEY75) Meyers, G. Reliable Software Through Composite Design, Petrocelli/Charte 1975.

(MIL72) H. D. Mills, "On the Statistical Validation of Computer Programs, FSC-72-6105, IBM Federal Systems Division, Gaithersburg, Md., 1972.

(MIL75) Miller, E. and R.A. Melton. "Automated Generation of Testcase Datasets", 1975 International Conference on Reliable Software, Los Angeles, April 1975.

(MIL81) Miller, E. and W. E. Howden. TUTORIAL: Software Testing and Validation Techniques. IEEE Computer Society Press, 1981.

(MYE76) Myers, G. Software Reliability: Principles and Practices. Wiley-Interscience, New York, 1976.

(MYE78) Myers. "A Controlled Experiment in Program Testing and Code Walkthroughs/Inspections." Communications of the ACM, September 1978.

(MYE79) Myers, G. The Art of Software Testing. Wiley-Interscience Publication, 1979.

(NAF72) Naftaly, S.M. and Cohen, M.C. "Test Data Generators and Debugging Systems", Workable Quality Control, Part I and II, Data Processing Digest, Vol.18, No.2 and 3, February and March, 1972.

(NBS80) "NBS Software Tools Database." dtd Oct 80. Available from the U. S. Dept. of Commerce, National Technical Information Service, 5285 Port Royal Road, Springfield, Virginia, 22151, Accession No. PB81-124935.

(NG78) Ng and Young. "A 1900 Fortran Post Mortem Dump System." Software Practice and Experience, July 1978.

(OST76) Osterweil, L.J. and L.D. Fosdick. "DAVE-A Validation Error Detection and Documentation System for Fortran Programs", Software Practice and Experience, 6, pp.473-486, Sept. 1976.

(OST78) Osterweil, L. J., J. R. Brown, and L. G. Stucki. "ASSET: A Lifecycle Verification and Visibility System." Proceedings of COMPSAC 78, IEEE Catalog No. 78CH1338-3C, November 1978, pp. 30-35.

(OTT79) Ottenstein, L.M. "Quantitative Estimates of Debugging Requirements", IEEE Transactions on Software Engineering, 1979, Vol.5, pp.504-514.

(PAI77) Paige. "Software Testing Principles and Practice Using a Testing Coverage Analyzer." Transactions of the Software '77 Conference, October 1977.

(PAN78) Panzl,D.J. "Automatic Software Test Drivers", IEEE Computer, April 1978.

(POW82A) Powell, P. B., editor. "Software Validation, Verification, and Testing Technique and Tool Reference Guide." NBS Special Publication 500-93, Superintendent of Documents, U.S. Govt. Printing Office, Wash, D.C. 20402, September 1982.

(POW82B) Powell, P. B., editor. "Planning for Software Validation, Verification, and Testing." NBS Special Publication 500-98, Superintendent of Documents, U.S. Govt. Printing Office, Wash, D.C. 20402, November 1982.

(PRE79) "Preliminary Ada Reference Manual", SIGPLAN Notices, Vol.14, No.6, part A, June 1979.

(PRO83) "Proceedings of the National Conference on Software Test and Evaluation." National Security Industrial Association, Software Group, 1-3 February 1983.

(QUA79) "Quantitative Software Models." Data and Analysis Center for Software, order No. SRR-1, RADC/ISISI (315) 336-0937, Autovon 587-3395.

(OUA83A) "Quality Metrics for Distributed Systems", Final report, Boeing Aerospace Company document, D-182-11377-1, -2, -3, 1983.

(QUA83B) "Quality Metrics Framework Enhancements for Software Aquisition" (CDRL A003), RADC contract F30602-82-C-0137 with Boeing Aerospace Company, July 1983.

(RAM75) Ramamoorthy, C.V. and K.H. Kim. "Software Monitors Aiding Systematic Testing and Their Optional Placement", Proceedings of the First National

Conference on Software Engineering, IEEE Catalog No. 75CH0992-8C, September, 1975.

(REI74) Reifer, D. J. and R. L. Ettenger. "Test Tools: Are They A Cure-All?". SAMSO-TR-75-13, October 1974.

(REI77) Reifer, D. J. and Trattner. "A Glossary of Software Tools and Techniques." IEEE Computer, July 1977.

(REI80) Reifer, D. J. and H. A. Montgomery. "Final Report, Software Tool Taxonomy." SMC-TR-004, 1 June 1980.

(REI) Reifer, D. J. "Software Tools Directory." Reifer Consultants Inc. 2733 Pacific Coast Highway, Suite 203, Torrance, CA 90505.

(RIC81) Richardson, D.J., L.A. Clarke. "A Partition Analysis Method to Increase Program Reliability". Proceedings of the Fifth International Conference of Software Engineering, 1981, pp.244-253.

(RUD77) B. Rudner, "Seeding/Tagging Estimation of Software Errors: Models and Estimates," RADC-TR-77-15, Polytechnic Institute of New York, 1977 (NTIS AD/A-036655).

(RYD75) Ryder, B.G.and A.D. Hall. "The PFORT Verifier", Computing Science Technical Report #12, Bell Laboratories, Murry Hill, New Jersey, March 1975.

(SAI82) Saib, S. H., et al. "Validation of Real-Time Software for Nuclear Plant Safety Applications." Electric Power Research Institute report EPRI NP-2646, Project 961 Final Report, November 1982.

(SCH73) Schaefer, M., A Mathematical Theory of Global Program Optimization. Prentice-Hall, Englewood Cliffs, N.J., 1973.

(SCH81) Schindler, M. "Today's Software Tools Point to Tomorrow's Tool Systems." Electronic Design, 23 July 1981, pp. 73-110.

(SCH79) Schneidewind and Hoffman. "An Experiment in Software Error Data Collection and Analysis." Transactions on Software Engineering, May 1979.

(SHN80) Shneiderman, B. Software Psychology-Human Factors in Computer and Information Systems. Winthrop Publishing, 1980.

(SMI76) Smith, P. "Fortran Code Auditor Users' Manual", RADC-TR-76-395, Vol. I, December 1976, NTIS accession no. A035-778.

(SMI79) Smith, C.U. and Browne, J.C. "Performance Specifications and Analysis of Software Designs", Proc. Conference on Simulation, Measurement and Modeling of Computer Systems, Boulder, CO., August 1979.

(SMI80) Smith, C.U. "The Prediction and Evaluation of the Performance of Software from Extended Design Specification", Ph.D. Dissertation, University of Texas at Austin, August 1980.

(SMI81) Smith, M. K. and D. R. Hudson, et al. "A Report on a Survey of Validation and Verification Standards and Practices at Selected Sites." Boeing Computer Services report BCS-40345, June 1981.

(SOF77) "Software Acquisition Management Guidebook: Validation and Certification", Air Force document, ESD-TR-77-326, 1977.

(SOF80) "Software Quality Metrics Enhancements", Final report, RADC-TR-80-109, 1980.

(SOF82) "Software Engineering Automated Tools Index." Software Research Associates, 1982.

(SOF83) "Software Interoperability and Reusability", Final report, Boeing Aerospace Company document, D182-11340-1, -2, 1983.

(SPE79) "Sperry Univac Series 1100 Fortran (ASCII) Programmer Reference," Sperry Rand Corporation, 1979.

(SPE82) "Specification of Software Quality Attributes", Interim reports, Boeing Aerospace Company documents, D182-11310-1, D182-11378-1, 1982.

(STA77) Stanfield and Skrukrud. "Software Aquisition Management Guidebook Software Maintenance Volume," System Development Corp., TM-5772/004/02, November 1977.

(STU73) Stucki, L.G. "Automatic Generation of Self-Metric Software", Proc. 1973 IEEE Symposium on Computer Software Reliability, 94(1973).

(STU75) Stucki, L. G. and G.L. Foshee. "New Assertion Concepts for Self-Metric Software", Proc. 1975 Conference on Reliable Software, pp.59-71.

(STU81) Stucki, Leon G. "Using ARGUS on EKS II." Boeing Computer Services, March 1981.

(SUK77) Sukert. "A Multi-Project Comparison of Software Reliability Models." Proceedings of the AIAA Conference on Computers in Aerospace, 1977.

(SUM) "Summary of Software Testing Measures." Software Research Associates report SRA TN-843.

(SYS77) "Systematic Software Development and Maintenance (SSDM)", Boeing Computer Services Document #10155, February 1977.

(TAY79) Taylor, R. N., R. L. Merilatt, and L. J. Osterweil. "Integrated Testing and Verification System for Research Flight Software." Boeing Computer Services report NAS1-15253, July 1979.

(TAY80) Taylor, R.N. "Assertions in Programming Languages", SIGPLAN Notices, Vol.15, No.1, January 1980, pp.105-114.

(TAY80B) Taylor, R.N. and L.J. Osteweil. "Anomaly Detection in Concurrent Software by Static Data Flow Analysis", IEEE Transactions on Software Engineering, Vol. SE-6, No. 3, pp. 265-278, May 1980.

(TEI72) Teichroew, D. "A Survey of Languages for Stating' Requirements for Computer-Based Information Systems", the University of Michigan, Proceedings of the Fall Joint Computer Conference, 1972, pp.1203-1224.

(TEI77) Teichroew, D. and E.A. Hershey, III. "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems", IEEE Transactions on Software Engineering, SE-3, 1977, (41-48).

(THA76) Thayer, T.A., et al. "Software Reliability Study." RADC report RADC-TR-76-238, August 1976.

(THR75) "THREADS: A Functional Approach to Project Control", Computer Sciences Corporation, El Segundo, CA, 1975.

(ULL73) Ullman, J.D. "Fast Algorithms for the elimination of common subexpressions". Acta Informatica, 2(1973), pp. 191-213.

(WEA78) "Weapon System Software Development." Military Standard MIL-STD-1679 (NAVY), 1 December 1978.

(WEI71) Weinberg, G.M. "Programming as a Social Activity", The Psychology of Computer Programming. Van Nostrand, Reinhold, 1971.

(WEI77) Weide, B. "A Survey of Analysis Techniques for Discrete Algorithm", Computing Surveys, Vol.9, No.4, Dec. 1977.

(WEI78) Weiss. "Evaluating Software Development by Error Analysis." Naval Research Lab NRL-8268, December 1978.

(WES79) Western District Utilities Manual, Boeing Computer Services Document #G0031 Rev. A, June 1979.

(WHI80) White, L.J. and E.I. Cohen. "A Domain Strategy for Computer Program Testing", IEEE Transactions on Software Engineering, Vol. SE-6, No.3, May 1980.

(WIN79) Winograd. "Beyond Programming Languages." Communication of the ACM, July 1979.

(WOO79) Woodfield. "An Experiment on Unit Increase in Program Complexity". IEEE Transactions on Software Engineering, March 1979.

(YEH77) Yeh, R.T., editor. Current Trends in Programming Methodology, Volume II. Prentice-Hall, Inc., 1977.

(YOU77) Yourdon, E. Structured Walk-throughs. Yourdon, Inc. 1977.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.


FILM